Step – by – Step Approach to the Solution

Step 1: Import Libraries

The code begins by importing the necessary libraries, including 'os' for interacting with the operating system, 'pandas' for working with data in tabular form, and various components from the Natural Language Toolkit (‘nltk’) for text processing. Additionally, the script downloads NLTK resources related to tokenization and stopwords.

Step 2: Define download_text_from_url Function.

This function is responsible for downloading content from a given URL.

Step 2.1: Use the requests library to make an HTTP request to the specified URL and get the response.

Step 2.2: Utilize BeautifulSoup to parse the HTML content of the webpage, making it easier to navigate and extract information.

Step 2.3: Extract the article title from the <title> tag of the HTML. The .string attribute is used to get the text content of the tag.

Step 2.4: Identify and extract paragraphs likely containing the article text. This is done using soup.find_all('p').

Step 2.5: Extract the text content from each paragraph and store it in the body_paragraphs list.

Step 2.6: Combine the article title and body paragraphs into a single string (article_text). Two newline characters (\n\n) are used to separate the title from the body.

Step 2.7: Construct the filename for the output text file. It is based on the url_id from the input DataFrame.

Step 2.8: Save the combined text content to a separate text file using the constructed filename.

Step 2.9: Return the filename to indicate the success of the download and saving process.

Step 2.10: If any exceptions occur during the process, handle them by printing an error message and returning None.

Overall, this function downloads content from a URL, extracts the article title and text, combines them into a formatted string, and saves the content to a text file named after the URL_ID.

Step 3: Define ‘calculate_text_metrics’ Function.

This function takes pre-tokenized words and sentences, performs several calculations related to text metrics, and returns the results. It calculates average sentence length, percentage of complex words, fog index, average words per sentence, syllables per word, count of personal pronouns, and average word length. The term "syllable" refers to a unit of sound containing a single vowel sound and usually consisting of one or more consonant sounds that precede or follow the vowel. It is a basic building block of a spoken word and is often used as a measure of the phonological complexity or "pronounceability" of a word.

Step 4: Define ‘syllable_count_word’ Function.

This function calculates the number of syllables in a word, considering different cases such as vowels and silent 'e'. It returns the maximum of 1 and the computed count. In the context of text analysis, counting syllables in a word is a way to estimate its pronunciation difficulty. In the provided code, the 'syllable_count_word' function is responsible for calculating the number of syllables in each word. This calculation is based on a simple heuristic that considers the presence of vowels and adjusts for silent 'e' at the end of words.

Step 5: Define ‘extract_and_analyze_text’ Function.

This function reads the contents of a text file, performs sentiment analysis using ‘TextBlob’, and calculates various text metrics using the previously defined ‘calculate_text_metrics’ function.

Open and Read File:

The function begins by opening the specified text file (‘file_path’) in read mode using with open(‘file_path’, 'r', encoding='utf-8') as file:. The with statement is used to ensure that the file is properly closed after reading.

Tokenization:

The ‘article_text’ is read from the file, and then the NLTK library is used for tokenization. Tokenization involves breaking down the text into sentences and words.

sentences = sent_tokenize(article_text): This line tokenizes the text into sentences.

words = word_tokenize(article_text): This line tokenizes the text into words.

Sentiment Analysis:

The ‘TextBlob’ library is utilized for sentiment analysis.

blob = TextBlob(article_text): A TextBlob object is created for the entire text.

polarity_score = blob.sentiment.polarity: The polarity score represents the positivity or negativity of the text.

subjectivity_score = blob.sentiment.subjectivity: The subjectivity score measures the subjectiveness of the text.

Text Metrics Calculation:

The calculate_text_metrics function is called to compute various text metrics. This function takes the list of words (words) and the list of sentences (sentences) as input.

Metrics include average sentence length, percentage of complex words, Fog Index, average words per sentence, syllables per word, count of personal pronouns, and average word length.

Construct Result Dictionary:

A dictionary (‘result_dict’) is constructed to store the extracted features and analysis results.

The dictionary includes fields such as 'URL_ID', 'URL', sentiment scores, various calculated metrics, and more.

Exception Handling:

The code is wrapped in a try-except block to handle any exceptions that may occur during file reading or analysis. If an exception occurs, an error message is printed, and None is returned.

Return Result:

The constructed result_dict is returned, containing all the information extracted and calculated from the text file.

Step 6: Define analyze_all_text_files Function.

This function iterates over rows in the input DataFrame, finds matching text files in the "Extracted_text" folder based on URL_ID, and analyzes each file using the extract_and_analyze_text function.

Step 7: Define main Function.

The main function readsfunction andta from an Excel file, analyzes text files using the analyze_all_text_files function, and saves the results to a CSV file.

Step 8: Run Main Script

The final block ensures that the main function is executed when the script is run.

This script, in essence, processes text files related to given URLs, performs sentiment analysis, calculates text metrics, and stores the results in a structured CSV format. The provided code is well-organized and modular, making it easier to understand and maintain.

Note: One of the URLs is invalid as that article in the URL does not exist, that URL alone cannot be extracted and analysed since that corresponding link does not exist.

Steps to run the Python file.

To run the provided Python code, follow these steps:

1. Install Required Libraries:

Make sure you have Python installed on your system.

Install the required libraries by running the following commands in your terminal or command prompt:

pip install pandas nltk textblob

Additionally, download the NLTK data by running the following Python code:

import nltk

nltk.download('punkt')

nltk.download('stopwords')

2. Prepare Input Data:

Place your input Excel file ("Input.xlsx") with the necessary information in the same directory as the Python script.

3. Create a Folder for Extracted Text:

Create a folder named "Extracted_text" in the same directory where the Python script is located. This folder should contain the text files to be analyzed.

4. Run the Script:

Open a terminal or command prompt.

Navigate to the directory containing the Python script and execute the following command:

python script_name.py

5. Check Output:

After running the script, it will generate an output CSV file ("output_analysis.csv") containing the analyzed results.

The CSV file will be created in the same directory as the Python script.

Dependencies:

For the code to run succesfully, the following are the dependencies:

1. Python:

Make sure you have Python installed on your system.

2. Libraries:

Install the required Python libraries using the following commands:

```
pip install pandas nltk textblob
```

3. NLTK Data:

After installing NLTK, download the necessary data by running the following Python code:

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```